ALGORITHMS DATA STRUCTURE

WEEK 1

Exercise 2: E-commerce Platform Search Function

CODE:

import java.util.\*;

class Product {

private int productId;

private String productName;

private String category;

public Product(int productId, String productName, String category) {

this.productId = productId;

this.productName = productName;

this.category = category;

}

public int getProductId() {

return productId;

}

public String getProductName() {

return productName.toLowerCase();

}

public String getCategory() {

return category.toLowerCase();

}

@Override

public String toString() {

return "ID: " + productId + ", Name: " + productName + ", Category: " + category;

}

}

public class Main {

public static Product searchById(Product[] products, int id) {

for (Product product : products) {

if (product.getProductId() == id) {

return product;

}

}

return null;

}

public static List<Product> searchByName(Product[] products, String name) {

List<Product> result = new ArrayList<>();

for (Product product : products) {

if (product.getProductName().contains(name.toLowerCase())) {

result.add(product);

}

}

return result;

}

public static List<Product> searchByCategory(Product[] products, String category) {

List<Product> result = new ArrayList<>();

for (Product product : products) {

if (product.getCategory().contains(category.toLowerCase())) {

result.add(product);

}

}

return result;

}

public static void main(String[] args) {

Product[] products = {

new Product(101, "Laptop", "Electronics"),

new Product(203, "Shoes", "Fashion"),

new Product(150, "Phone", "Electronics"),

new Product(120, "Watch", "Accessories"),

new Product(180, "Tablet", "Electronics")

}

Scanner sc = new Scanner(System.in);

System.out.println("====== Product Search Menu ======");

System.out.println("1. Search by Product ID");

System.out.println("2. Search by Product Name");

System.out.println("3. Search by Category");

System.out.print("Enter your choice (1-3): ");

int choice = sc.nextInt();

sc.nextLine();

switch (choice) {

case 1:

System.out.print("Enter Product ID: ");

int id = sc.nextInt();

Product found = searchById(products, id);

System.out.println(found != null ? found : "Product not found.");

break;

case 2:

System.out.print("Enter Product Name (partial or full): ");

String name = sc.nextLine();

List<Product> byName = searchByName(products, name);

if (!byName.isEmpty()) {

byName.forEach(System.out::println);

} else {

System.out.println("No products found with name: " + name);

}

break;

case 3:

System.out.print("Enter Product Category (partial or full): ");

String category = sc.nextLine();

List<Product> byCategory = searchByCategory(products, category);

if (!byCategory.isEmpty()) {

byCategory.forEach(System.out::println);

} else {

System.out.println("No products found in category: " + category);

}

break;

default:

System.out.println("Invalid choice. Please select 1, 2, or 3.");

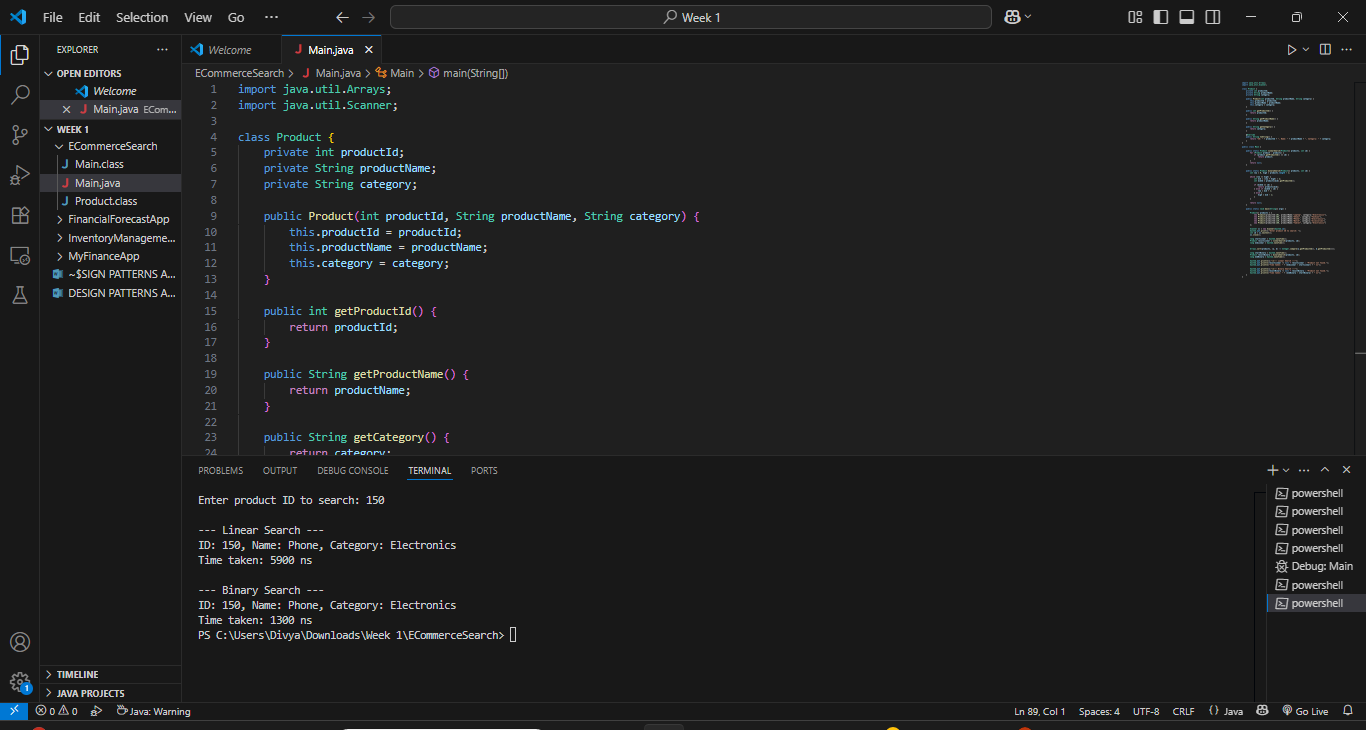
}

sc.close();

}

}

OUTPUT:



Exercise 7: Financial Forecasting

CODE:

public class FinancialForecast {

public static double forecastValue(double initialValue, double growthRate, int years) {

if (years == 0) return initialValue;

return forecastValue(initialValue, growthRate, years - 1) \* (1 + growthRate);

}

public static void main(String[] args) {

double initialValue = 10000;

double growthRate = 0.08;

int years = 10;

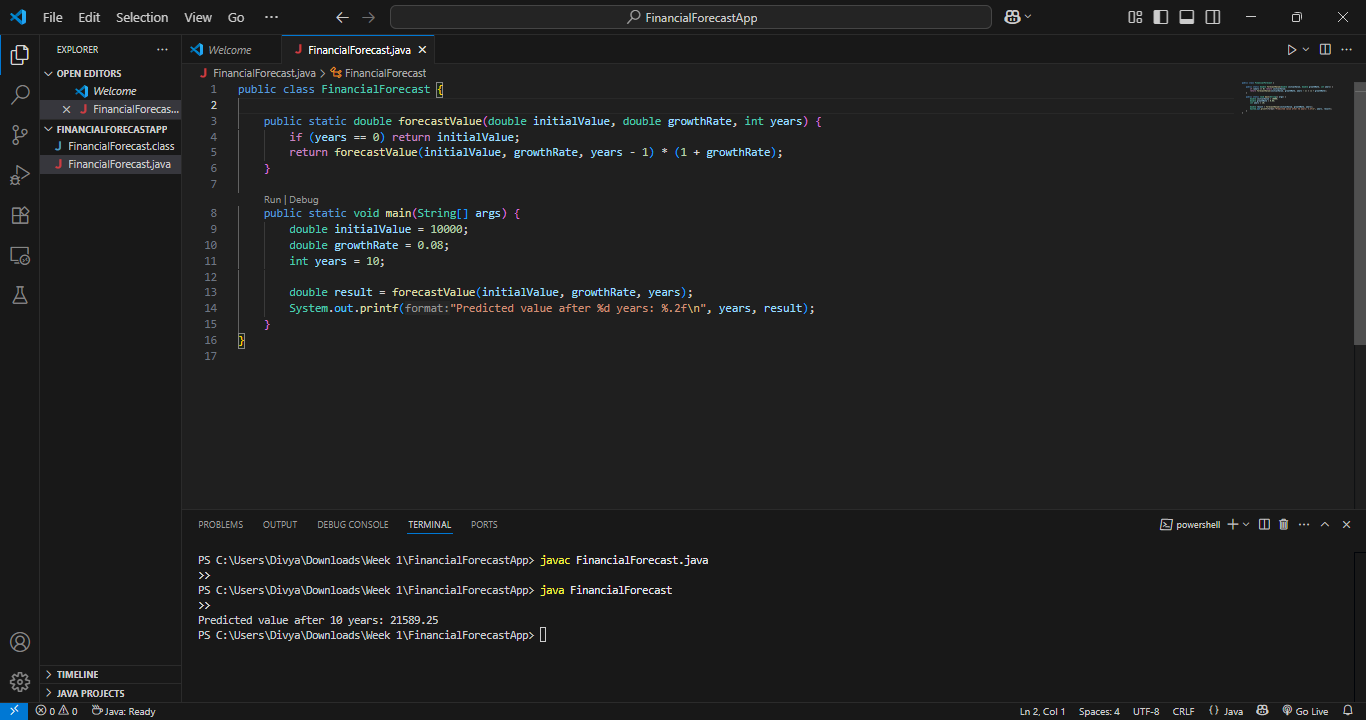
double result = forecastValue(initialValue, growthRate, years);

System.out.printf("Predicted value after %d years: %.2f\n", years, result);

}

}

OUTPUT:



DESIGN PATTERNS AND PRINCIPLES

WEEK 1

Exercise 1: Implementing the Singleton Pattern

CODE:

package singleton;

public class Logger {

private static Logger instance;

private Logger() {

System.out.println("Logger initialized");

}

public static Logger getInstance() {

if(instance==null) {

instance=new Logger();

}

return instance;

}

public void log(String message) {

System.out.println("Log: "+message);

}

}

TEST CLASS:

package singleton;

public class SingletonTest {

public static void main(String args[]) {

Logger logger1=Logger.getInstance();

Logger logger2=Logger.getInstance();

logger1.log("First message");

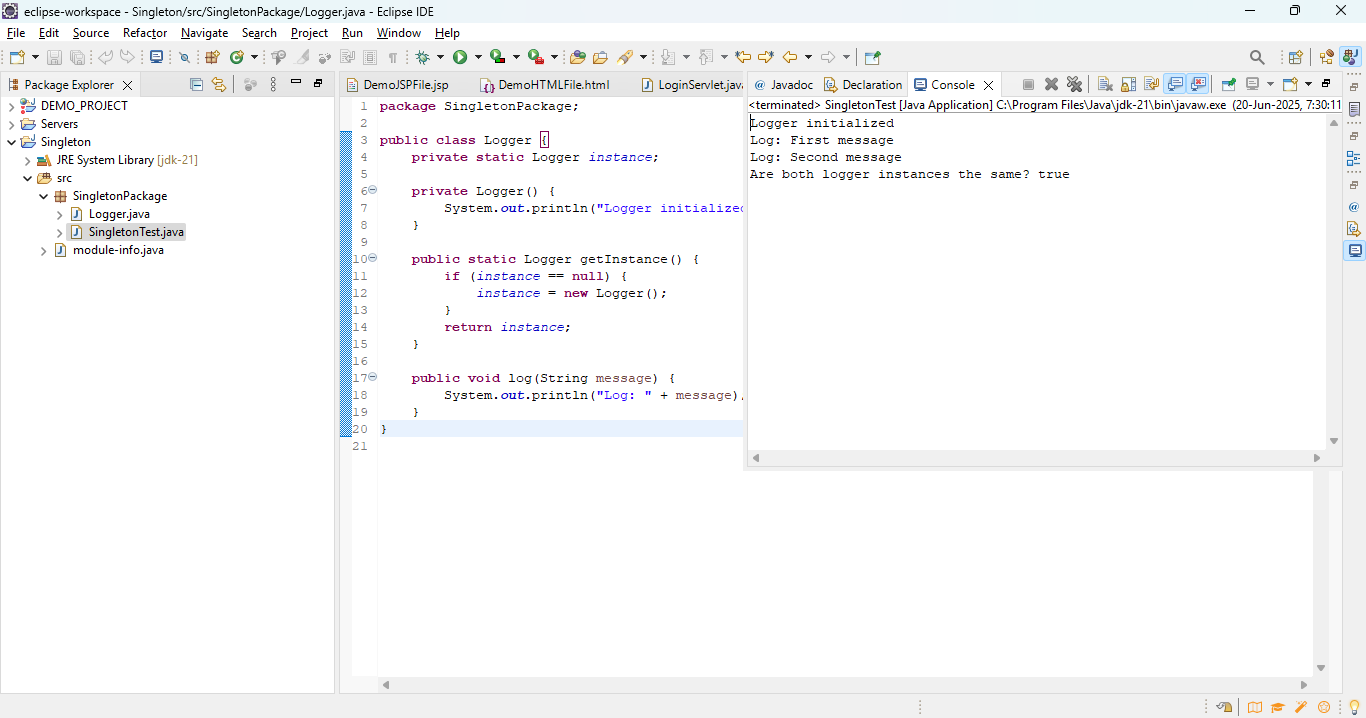
logger2.log("Second message");

System.out.println("Are both logger instances the same? " + (logger1 == logger2));

}

}

OUTPUT:



Exercise 2: Implementing the Factory Method Pattern

CODE:

Document.java

package factoryMethod;

public interface Document {

void open();

}

class WordDocument implements Document {

public void open() {

System.out.println("Word Document Created");

}

}

class PdfDocument implements Document{

public void open() {

System.out.println("Pdf Document Created");

}

}

class ExcelDocument implements Document{

public void open() {

System.out.println("Excel Document Created");

}

}

DocumentFactory.java

package factoryMethod;

abstract class DocumentFactory{

abstract Document createDocument();

}

class WordDocumentFactory extends DocumentFactory {

Document createDocument() {

return new WordDocument();

}

}

class PdfDocumentFactory extends DocumentFactory{

Document createDocument() {

return new PdfDocument();

}

}

class ExcelDocumentFactory extends DocumentFactory{

Document createDocument() {

return new ExcelDocument();

}

}

DocumentFactoryTest.java

package factoryMethod;

public class DocumentFactoryTest {

public static void main(String args[]) {

DocumentFactory docW=new WordDocumentFactory();

docW.createDocument().open();

DocumentFactory docP=new PdfDocumentFactory();

docP.createDocument().open();

DocumentFactory docE=new ExcelDocumentFactory();

docE.createDocument().open();

}

}

OUTPUT:

